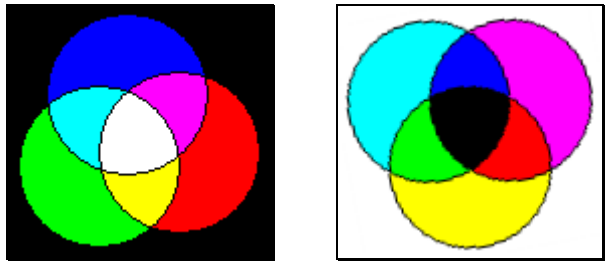
3.5 Spaţii de culoare (RGB, CIE XYZ, CIE Lab, CIE Luv)

3.5.1 Sistemul RGB

Modelul RGB este un model aditiv care presupune că plecând de la negru (fără culoare) prin adăugarea celor trei culori de bază (primare de la cele trei surse de lumină ***R**ed, **G**reen, **B**lue*) se obţin celelalte nuanţe până la alb (vezi Figura 18 a). Prin combinarea a două dintre acestea în proporţie egală se obţin culorile aditive secundare (complemenatre - ***C**yan, **Y**ellow, **M**agenta*). Într-un model substractiv culorile de bază primare sunt ***C**yan, **Y**ellow* şi ***M**agenta*. În acest model se presupune că se pleacă de la alb (absenţa culorilor), iar prin scăderea (ştergerea) culorilor se ajunge la negru (vezi Figura 18 b).



a) b)

Figura 18 – Sistemul RGB / CYM

În sistemul RGB reprezentăm o culoare prin trei coordonate (numere naturale) reprezentând cantităţile culorilor de bază, roşu, verde şi albastru (***R**ed, **G**reen, **B**lue*), deci în spaţiul ***RGB*** (vezi Figura 19) sau cel complementar (***C**yan, **Y**ellow, **M**agenta*). Utilizând opt bit pentru reprezentarea fiecărei coordonate (valori a componentei ***red**, **green*** sau ***blue***) vom obţine un spaţiu al culorilor cu valori din domeniul $[0..255]^3$, adică 2^{24} valori distincte, deci peste 16 milioane de nuanţe.

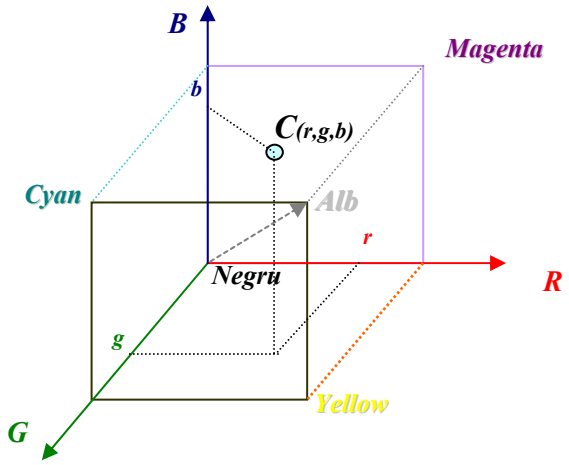


Figura 19 - Spaţiul culorilor în modelul ***RGB***

Spectrul este o bandă continuă de lungimi de undă emisă, reflectată sau transmisă de deverse obiecte. Ochiul uman nu poate vedea întregul spectru, doar culori cu valori din domeniul [400nm-700nm] corespunzător [**albastru-roşu**]. În practică, spectrul este reprezentat ca un vector de numere care măsoară intensitatea diferitelor lungimi de undă. Usual, un spectrofotometru are o rezoluţie de 1-3 nm, măsurând de la 100 la 300 de valori pentru a măsura domeniul vizibil. Presupunând că un aparat foarte simplu scanează un spectru la fiecare 3 nm (100 de puncte pentru 300 nm) şi măsoară fiecare punct cu o precizie de doar 8 bit. Aceasta înseamnă că aparatul poate distinge mai mult de 10240 de spectre diferite (atâtea numere pot fi codificate pe 8 bit x 100 puncte = 800 bit de date). Dar oamenii pot distinge doar în jur de 10 milioane de culori. Aceasta se întâmplă pentru că ochiul are doar trei tipuri de conuri receptoare în retină care răspund diferitelor lungimi de undă a luminii. Astfel, creierul receptează doar trei semnale pentru fiecare spectru, acestea dându-ne percepţia culorii.

3.5.2 Interpretări colorimetrice computerizate

Un sistem tricromatic poate defini orice culoare combinaţia a trei culori de bază în cantitati unic precizate (culorile fundamentale sunt alese astfel încât nici una să nu poatăfi obţinută prin amestecul celorlaltor două). Cel mai cunoscut sistem tricromatic este sistemul RVI (R-rosu, V-verde, I-indigo) caracterizat prin cele trei culori de bază având următoarele lungimi de undă: rosu λ =700 nm; verde λ =546,1 nm; indigo λ = 435 nm.

Orice culoare monocromatică spectrală, într-un sistem tricromatic este caracterizată prin trei valori numite valori spectrale sau coeficienţi de distribuţie. În sistemul RVI, există culori care sunt caracterizate şi prin coeficienţi negativi (coordonate negative ale culorilor de bază). Pentru a elimina valorile negative la precizarea unei culori, s-au definit trei culori virtuale (notate cu X, Y, Z) care nu mai prezintă inconvenientul precizat. Aceste culori constitue valorile fundamentale ale culorii în sistemul tricromatic CIE (Commission Internationale De L'Eclairage), fiind denumite şi coordonate tricromatice sau valori tristimu.

3.5.3 Caracterizarea şi măsurarea culorilor în sisteme tricromatice.

În sistemul CIE, o culoare poate fi reprezentată grafic într-un sistem tridimensional prin punctul de coordonate (X,Y,Z). Totalitatea punctelor ce corespund tuturor culorilor posibile constituie *spaţiul culorilor* (vezi **Error! Reference source not found.** Figura 20).

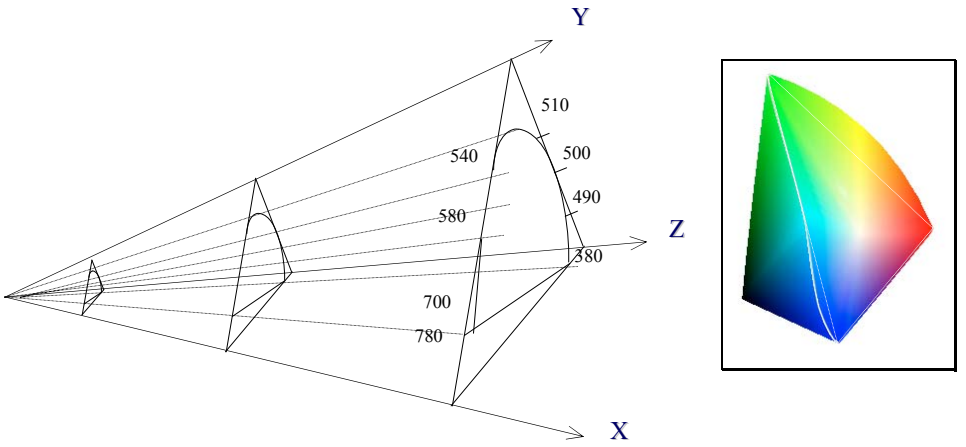


Figura 20 - Spaţiul culorilor în sistemul de coordonate CIE XYZ

Măsurarea culorilor în sistemul CIE constă în determinarea valorilor X,Y,Z. În acest scop se consideră cei trei factori care caracterizează culoarea, şi anume:

- *iluminantul*, reprezentat prin ***curba de distribuţie a energiei*** E(λ);
- *corpul colorat*, reprezentat prin ***curbele de remisiune*** **(*reflexie*)** R(λ);
- *ochiul omenesc* caracterizat prin ***curbele de distribuţie spectrală*** pentru observatorul normal $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, $\bar{z}(\lambda)$.

Pentru o lungime de undă λ, ***energia transmisă de corpul colorat*** este egală cu produsul E(λ)·R(λ).

Energia transmisă pentru toate lungimile de undă ale spectrului vizibil (φ) este egală cu $\sum_{\lambda} E(\lambda)\cdot R(\lambda)$, şi reprezintă *interacţiunea iluminant-corp*.

Pentru a se determina interacţiunea iluminant-corp colorat-ochi, funcţia φ trebuie înmulţită cu fiecare din cele trei funcţii $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, $\bar{z}(\lambda)$.

Se obţin astfel pentru X, Y, şi Z următoarele formule de calcul:

$$X = \sum_{\lambda=380}^{770} E(\lambda) \cdot R(\lambda) \cdot \bar{x}(\lambda)$$

$$Y = \sum_{\lambda=380}^{770} E(\lambda) \cdot R(\lambda) \cdot \bar{y}(\lambda)$$

$$Z = \sum_{\lambda=380}^{770} E(\lambda) \cdot R(\lambda) \cdot \bar{z}(\lambda)$$

Pentru calcule, mărimile $E(\lambda) \cdot \bar{x}(\lambda)$; $E(\lambda) \cdot \bar{y}(\lambda)$; $E(\lambda) \cdot \bar{z}(\lambda)$, sunt normate şi înscrise în tabele, astfel încât la o măsurare rămâne de determinat doar curba de remisiune $R(\lambda)$.

Pentru sistemul CIELAB 76, coordonatele tricromatice X, Y, Z sunt transformate în alte trei coordonate:

L^* (*coordonata luminozitate*),
a^* (*coordonata roşu-verde*) şi
b^* (*coordonata galben-albastru*) :

$$\begin{cases} L^* = 116 \cdot f(Y/Y_0); \\ a^* = 500 \cdot [f(X/X_0) - f(Y/Y_0)] \\ b^* = 200 \cdot [f(Y/Y_0) - f(Z/Z_0)] \end{cases}$$

unde X_0, Y_0 şi Z_0 sunt coordonatele tricromatice care se obţin în cazul unui material cu suprafaţă reflectantă perfect albă:

$$\begin{cases} X_0 = \sum_{\lambda=380}^{770} E(\lambda) \cdot \bar{x}(\lambda) \\ Y_0 = \sum_{\lambda=380}^{770} E(\lambda) \cdot \bar{y}(\lambda) \\ Z_0 = \sum_{\lambda=380}^{770} E(\lambda) \cdot \bar{z}(\lambda) \end{cases}$$

iar $f(\Phi) = \Phi^{1/3} - 16/116$.

Vizualizarea culorilor în sistemul CIE Lab se poate vedea în Figura 21. În *adâncime* este masurată luminozitatea, iar pe *orizontala* rosu-verde, iar pe *verticală* galben-albastru:

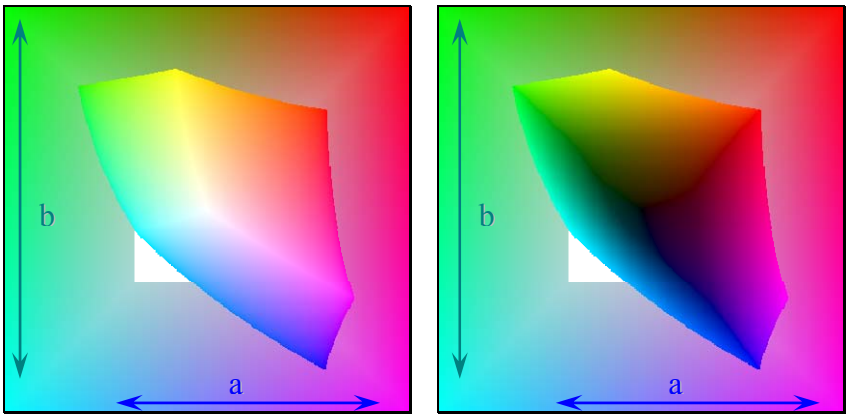


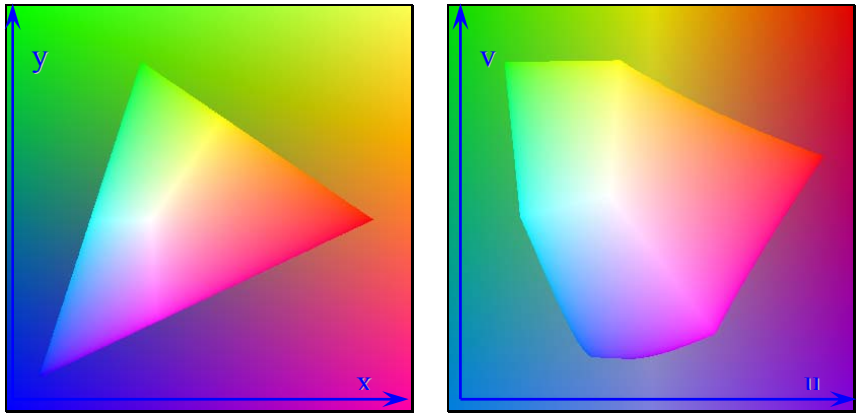
Figura 21 - CIE L a*b* văzut dinspre *alb*, respectiv dinspre *negru*

Diferenţa de culoare *ΔE** dintre două culori (probe) reprezintă distanţa geometrică (*Euclidiană*) dintre punctele corespunzătoare (celor două probe) din spaţiul culorilor. În sistemul **CIELAB**, aceasta se calculează cu ajutorul formulei:

$$\Delta E^* = (\Delta L^{*2} + \Delta a^{*2} + \Delta b^{*2})^{1/2}$$

unde:

ΔL* reprezintă diferenţa de *luminozitate* dintre cele două probe,
Δa* reprezintă diferenţa dintre coordonatele *roşu-verde*,
Δb* reprezintă diferenţa dintre coordonatele *galben-albastru*.



a) Δ CIE XYZ (1931) b) CIE L u'v' (1976)

Figura 22 - Spatiul CIE al culorilor vizibile.

3.5.4 Conversia RGB → CIE-Lab

Trecerea de la modelul RGB la CIE Lab se realizează prin intermediul spaţiului XYZ în două etape astfel:

a) RGB → CIE XYZ:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

b) XYZ → CIE Lab:

$$L^* = 116 * (Y/Y_n)^{1/3} - 16 \quad \text{pentru } Y/Y_n > 0.008856$$
$$903.3 * Y/Yn \quad \text{în rest}$$
$$a^* = 500 * (f(X/X_n) - f(Y/Y_n))$$
$$b^* = 200 * (f(Y/Y_n) - f(Z/Z_n))$$

unde $f(t) = t^{1/3}$ pentru $t > 0.008856$

$7.787 * t + 16/116$ în rest

3.5.5 Conversia CIE-Lab → RGB

Trecerea de la sistemul CIE Lab la modelul RGB la se tot prin spaţiul intermediar XYZ:

a) CIE Lab → XYZ :

$$X = X_n * (P + a^*/500)^3$$
$$Y = Y_n * P^3$$
$$Z = Z_n * (P - b^*/200)^3$$

unde $P = (L^* + 16) / 116$

b) CIE XYZ → RGB astfel:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

3.5.6 Conversia RGB → CIE-Luv

După conversia din spaţiul RGB în XYZ (descrisă mai sus la 3.5.4 a) se va efectua conversia în spaţiul CIE Luv (Figura 22) astfel:

$$L = 116 * (Y/Y_n)^{1/3} - 16$$
$$U = 13 * L * (u - u_n)$$
$$V = 13 * L * (v - v_n)$$

unde

$$u = 4*X / (X + 15*Y + 3*Z)$$
$$v = 9*Y / (X + 15*Y + 3*Z)$$
$$u_n = 4*x_n / (-2*x_n + 12*y_n + 3)$$
$$v_n = 9*y_n / (-2*x_n + 12*y_n + 3)$$

3.5.7 Conversia CIE-Luv → RGB

Mai întâi vom efectua conversia din spaţiul CIE Luv în spaţiul XYZ:

$$Y = Y_n * ((L+16)/116)^3$$
$$X = -9*Y*u/((u-4)*v - u*v)$$
$$Z = (9*Y - 15*v*Y - v*X)/3*v$$

unde

$$u = U / (13*L) + u_n$$
$$v = V / (13*L) + v_n$$

apoi se va efectua conversia în spaţiul RGB (descrisă mai sus la 3.5.5 b).

3.6 Creşterea realismului imaginilor tridimensionale

Acest paragraf prezintă câteva metode de îmbunătăţire a imaginilor în sensul apropierii calităţii lor de imaginile reale. Dintre aceste metode prezentate în literatura de specialitate (*eliminarea suprafeţelor şi muchiilor acoperite* pentru extragerea elementelor de frontieră ascunse, *perspectiva* pentru informaţiile de profunzime, *proiecţiile dinamice* pentru reprezentarea obiectelor în mişcare, *indici de intensitate* sau *variaţia de culoare* utilizate pentru modificarea culorilor din adâncime, *texturi şi detalii de suprafaţă* pentru reprezentarea microstructurilor feţelor, *secţionarea* cu un plan frontal utilizată la vizualizarea interiorului obiectului, *iluminarea curpurilor* prin utilizarea luminilor şi umbrelor şi *stereografia* pentru redarea în *relief* a a

obiectelor tridimensionale, dintre care am ales doar câteva pe care le-am considerat mai importante (vezi [[7,11, 29,63]).

3.6.1 Eliminarea suprafeţelor acoperite

Algoritmii de eliminare a zonelor *invizibile* sunt în general mari consumatori de resurse (memorie şi timp). Există trei clase de algoritmi şi anume:

a) *Algoritmi spaţiu-imagine* prin care se compară fiecare pixel cu fiecare faţă, deci de complexitate P*F (P=numărul de pixeli din fereastră, F=numărul de feţe ale corpului) pentru a decide cărei feţe îi aparţine fiecare pixel în scopul colorării corespunzătoare;

b) *Algoritmi spaţiu-obiect* care compară feţele două câte două, deci de complexitate F*F, pentru a decide relaţia de acoperire dintre feţe în scopul determinării unei ordini de transpunere a feţelor pe ecran astfel încât feţele mai îndepărtate se transpun primele iar cele mai apropiate vor fi desenate ultimele (se vor acoperi una pe alta).

c) *Algoritmi hibrizi* care utilizează elemente convenabile din primele două clase.

Pentru simplificarea comparaţiilor dintre feţe sunt utilizate diverse tipuri de *ferestre ecran* (unidimensionale, bidimensionale sau chiar tridimensionale) care conţin feţele studiate şi pentru care comparaţiile sunt mult mai uşoare (vezi Figura 23).

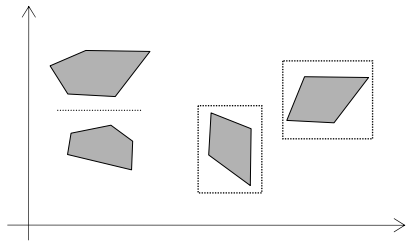


Figura 23 – Compararea feţelor

Dintre algoritmii mai cunoscuţi (prezentaţi în [[7]) amintim următorii:

- *Buffer de adâncime (Z-buffer) ,*
- *Linie de baleiaj ,*
- *Sortarea în adâncime (depth-sort)* şi
- *Subdivizarea ariilor în arii elementare disjuncte*.

- ***Algoritmul Z-buffer***

Algoritmul utilizează o matrice D care conţine pentru fiecare pixel P_{ij} cota z a punctului reprezentat prin P_{ij} (în urma proiecţiei) şi notată cu $z(P_{ij})$. Presupunem că observatorul se află pe axa Oz, aşa cum am văzut în paragraful precedent.

Algoritmul Z-Buffer este:
Pentru fiecare pixel P_{ij} din fereastra ecran execută
Colorează (P_{ij}, Culoarea_de_fond);
$D_{i,j}$:= minimă, negativă (minus infinit);
Pentru fiecare faţă f a obiectului execută
Pentru fiecare pixel P_{ij} corespunzător feţei f execută
Dacă $z(P_{ij}) > D_{ij}$ Atunci { (*) }
Colorează (P_{ij}, Culoarea_feţei f);
$D_{i,j}$:= $z(P_{ij})$;
Sf_Z_Buffer.

Datorită faptului că acest algoritm utilizează multă memorie pentru a reprezenta matricea D, acesta se poate modifica pentru a rezolva această problemă pentru o linie sau pentru o fereastră ecran.

Pentru a micşora durata de execuţie a algoritmului prezentat, calculele de distanţă ($z(P_{ij})$) pot fi evitate prin determinarea ecuaţiei planului care conţine fiecare faţă şi determinarea recurentă a cotei z aşa cum se poate vedea în Figura 24. Pentru trei puncte ale unei feţe alese se poate determina ecuaţia planului ce conţine faţa : Ax+By+Cz+D=0.

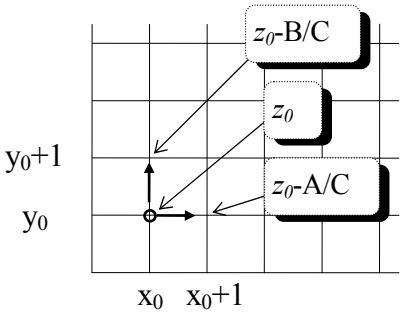


Figura 24 – Calcul z

Odată calculată cota z_0 a unui pixel $P_{x_o y_o}$, se deduce uşor că pentru punctele vecine cotele sunt egale cu z_0-A/C respectiv z_0-B/C.

Acest algoritm, prezentat pentru eliminarea feţelor ascunse, se poate uşor adapta pentru eliminarea muchiilor ascunse şi de asemenea se poate extinde pentru feţe cu un anumit grad de transparenţă, modficând condiţia de *vizibilitate* (*) din algoritm.

- ***Algoritmul Linie de baleiaj***

Acest algoritm este o extindere a algoritmului de umplere a interiorului uni poligon şi poate fi utilizat în cazul reprezentărilor poliedrale, deoarece feţele în acest caz sunt poligoane.

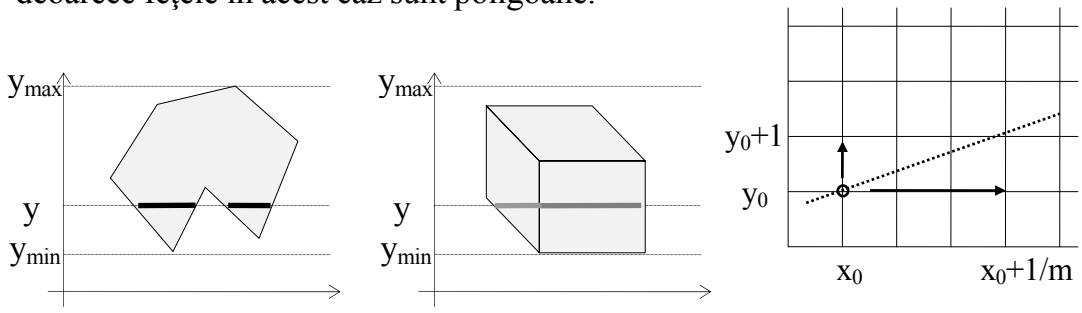


Figura 25 – *Linie de baleiaj*

Algoritmul de umplere (Figura 25-a,b) este:

Pentru fiecare linie de baleiaj y := ymin, ymax execută
Determină intersecţiile x_i cu fiecare latură;
Ordonează crescător şirul x_i ;
Pentru fiecare pereche de coordonate determinate
Trasează segmente orizintale de la x_{2*k-1} la x_{2*k}.

Pentru a nu mai calcula de fiecare dată intersecţiile linie de baleiaj cu laturile poligonului se poate utiliza o *listă* (ordonată după x_{min}) *de segmente active* (notată cu *Lsa* care conţine laturile intersectate la un moment dat de linia de baleiaj) fiecare conţinând informaţiile :

y_{min}, x_{min}, y_{max}, $1/m$ (*m*-reprezintă panta dreptei, iar 1/m creşterea pe x corespunzătoare creşterii cu o unitate pe y, vezi figura (Figura 25-c) .

Lsa:=∅;
Pentru fiecare linie de baleiaj y := ymin, ymax execută
Dacă ∃ o latură s a poligonului pentru care y=y_{min} *Atunci Lsa:=Lsa∪{s};*
Intersecţiile x_i *sunt valorile* x_{min} din *Lsa;*
Pentru fiecare pereche de coordonate x_i
Trasează segmente orizintale de la x_{2*k-1} *la* x_{2*k} *;*
Pentru fiecare segment s∈Lsa execută
$x_{min} := x_{min} + 1/m$;
Dacă y=y_{min} *Atunci Lsa:=Lsa \ {s};*

Pentru algoritmul de eliminare a feţelor acoperite vom mai reţine în lista de segmente şi feţele care formează fiecare muchie. Pentru fiecare segment orizontal desenat se va determina care este faţa cea mai apropiată care îl conţine şi se va colora în culoarea feţei respective până la următoarea intersecţie. Dacă în listă nu mai apar segmente, atunci ordinea rămâne aceeaşi, deci nu se mai determină faţă cea mai apropiată. Este de asemenea utilă o informaţie binară pentru fiecare faţă care ne va spune dacă linia de baleiaj intră sau părăseşte faţa respectivă pentru a ounaşte care feţe sunt active şi trebuie luate în considerare la determinarea culorii de desenare.

- ***Algoritmul depth-sort***

Algoritmul constă în ordonarea feţelor astefl încât cea mai depărtată de observator se transpune prima iar cea mai apropiată ultima. In felul acesta feţele se vor acoperi efectiv de către feţele mai apropiate.

Pentru aceasta se vor efectua următoarele:

- Se ordonează feţele după valoarea z maximă;
- Se transpun prin baleiaj feţele în ordinea determinată.

Pentru a determina această ordine va trebui să determinăm o relaţie de acoperite între feţe, după care putem să efectuăm ordonarea.

Două feţe F_i şi F_j pot fi în următoarele situaţii:

I. F_i *Acoperă* F_j $(F_i \supset F_j)$;

II. F_i *Acoperită de* F_j $(F_i \subset F_j)$;

III. F_i , F_j *Disjuncte* $(F_i \cap F_j = \varnothing)$;

IV. F_i , F_j *Se intersectează* $(F_i \cap F_j \neq \varnothing)$.

În prime trei situaţii putem rezolva problema colorării, iar în situaţia IV se vor împărţi feţele astfel încât putem reduce problema la unul din cele trei cazuri favorabile (I-III).

În algorim vom utiliza o *matrice de acoperire* în care un element M_{ij} = 1 dacă faţa *i* acoperă faţa *j* (I) şi $M_{ij} = 0$ în rest. *Coeficientul de acoperire* A_i ne va da numărul de feţe *acoperite* de faţa *i*. A_i se poate calcula ca suma elementelor de pe linia *i* din matricea de acoperire M.

Algoritmul este următorul:

Cât timp $\exists$ *feţe* F_i *pentru care* $A_i \geq 0$ *execută*
i:=Alege(F_i) *{ pentru care* $A_i=0$ *};*
Transpune (F_i); A_i=-1*;*
Pentru toate feţele F_j *care acoperă faţa* F_i *decrementează* A_j*;*

Pentru a determina relaţiile dintre două feţe F_i şi F_j se procedează după cum urmează:

- Se compară extinderile ecran şi dacă acestea nu se suprapun atunci feţele vor fi în relaţia III;
- Se determină ecuaţia planului ce conţine faţa F_i apoi se verifică dacă faţa F_j şi observatorul sunt de acceaşi parte a planului, caz în care suntem în situaţia II, altfel în situaţia I. Se procedează analog pentru a determina ecuaţia planului ce conţine faţa F_j apoi se verifică dacă faţa F_i şi observatorul sunt de acceaşi parte a planului determinat, etc.;
- Pentru zona comună se poate analiza chiar fiecare pixel pentru a stabili faţă cea mai apropiată.

- ***Algoritmul prin subdivizare a ariilor***

Această metodă compară zone dreptunghiulare ale ferestrei (numite *arii elementare* notate în cele ce urmează cu A_E) cu proiecţii ale feţelor corpului. Relaţiile dintre o arie elementară A_E şi o faţă F_i următoarele (vezi Figura 26):

I. A_E *Conţine* F_i $(A_E \supset F_i)$;
II. A_E *Conţinută în* F_i $(A_E \subset F_i)$;
III. A_E , F_i *Disjuncte* $(A_E \cap F_i = \varnothing)$;
IV. A_E , F_i *Se intersectează parţial* $(A_E \cap F_i \neq \varnothing)$.

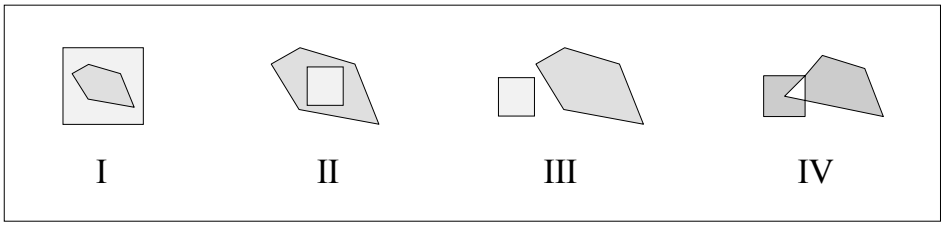


Figura 26 – Poziţii relative ale ariilor elementare

Desenarea conţinutului unei arii elementare se realizează astfel:

1) Dacă $A_E \cap F_i = \varnothing$ (III), $\forall F_i$ atunci A_E se colorează cu culoarea de fond;
2) Dacă există o singură faţă F_i pentru care $A_E \cap F_i \neq \varnothing$ (IV), atunci exteriorul $(A_E \setminus F_i)$ se colorează cu culoarea de fond, iar interiorul $(A_E \cap F_i)$ cu culoarea feţei F_i;
3) Dacă există o singură faţă F_i pentru care $A_E \subset F_i \neq \varnothing$ (I), atunci A_E se colorează cu culoarea feţei F_i;
4) Dacă există mai multe feţe F_i pentru care avem $A_E \cap F_i \neq \varnothing$ (IV) dar există o singură faţă F_j pentru care $A_E \subset F_j$ şi F_j este cea mai apropiată, atunci A_E se colorează cu culoarea feţei F_j;
5) Dacă nu se poate aplica nici o regulă dintre primele patru descrise, atunci se împarte A_E în subarii (de exemplu în patru, vezi Figura 27) şi se verifică

din nou condiţiile 1) - 4). Dacă nici atunci nu sunt verificate, se împarte din nou, şi tot aşa până se ajunge la arie egală cu un pixel, situaţie în care preblema poate fi rezolvată.

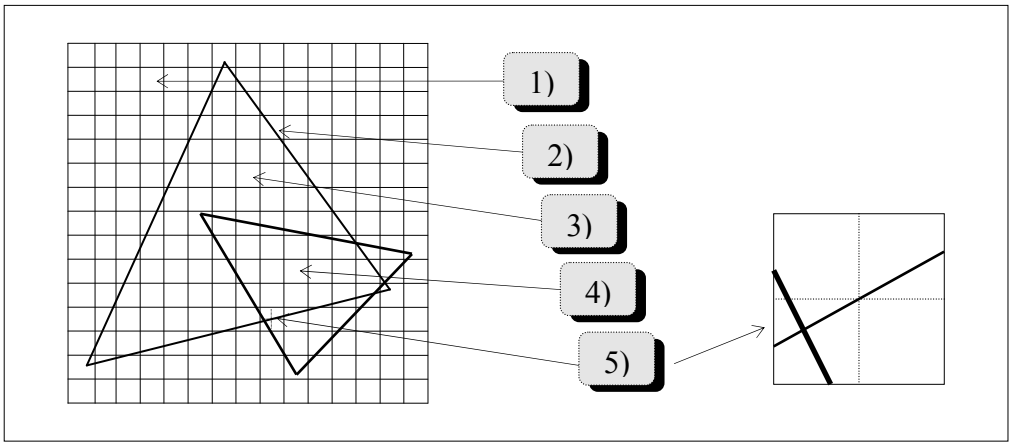


Figura 27 – Împărţirea în subarii

3.6.2 Texturi

Utilizarea texturilor în grafica tridimensională are un rol important în creşterea realismului imaginilor (mai ales dacă acestea prezintă mici imperfecţiuni, adică defecte intenţionat introduse), deoarece aceste detalii de pe suprafaţa obiectelor dau multe informaţii (material, poziţie, dimensiuni, etc.) despre corpul reprezentat. Există trei categorii de texturi şi anume:

a) Constante ca mărime şi orientare pe suprafaţa corpului,
b) Variabile ca mărime şi orientare în funcţie de poziţia feţelor,
c) Neregulate aleatoare ca mărime şi orientare (fractali).

a) Texturile constante

Acestea se utilizează atât în grafica 2-D cât şi în grafica 3-D pentru tapetarea fondurilor scenelor. Acestea se pot defini printr-o matrice de culori sau în sistem vectorial prin coodonate relative ale capetelor segmentelor descrise. Există biblioteci de texturi caracteristice diferitelor obiecte sau materiale cum ar fi: *fagure, lemn, parchet, sticlă, apă, iarbă,* etc.

Aplicarea unei texturi se poate face la umplere, unde un punct P_{ij} se va colora ţinând cont de o textură defintă prin matricea T cu *m* linii şi *n* coloane numerotate de la 0 la *m*-1 respectiv *n*-1 în culoarea $T_{i \ Mod \ m \ , \ j \ Mod \ n}$ (vezi Figura 28).

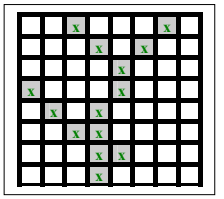


Figura 28 – Textură constantă

O aplicaţie interesantă şi utilă a texturilor constante o reprezintă utilizarea lor în simularea culorilor combinate, de exemplu dacă descriem un model asemănător cu cel al pătrăţelelor de pe tabla de şah folosind două culori: alb şi galben, vom genera o culoare de umplere între cele două (o nuanţă de galben deschis). În definirea unui astfel de model trebuie evitate haşurile.

a) Texturile variabile

Un exemplu des utilizat de texturi variabile îl constituie aplicarea textelor pe feţele corpurilor (simboluri scrise pe suprafaţa obiectelor).

Pentru aceasta este nevoie de o *digitizare* a caracterelor prin segmente sau prin linii poligonale închise (vezi Figura 29). Există o adevărată industrie de fonturi care crează biblioteci de caractere .

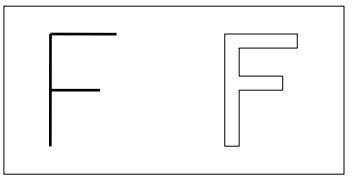


Figura 29 - Textură variabilă

Aplicarea pe faţa dorită a unei texturi variabile se realizează în trei etape:

- Se aduce caracterul pe faţa dorită prin două rotaţii (într-un plan paralel cu faţa pe care se doreşte aplicarea) apoi printr-o translaţie (Figura 30);
- Se proiectează punctele critice care descriu caracterul odată cu obiectul;
- Se construieşte textura unind punctele critice sau prin umplere.

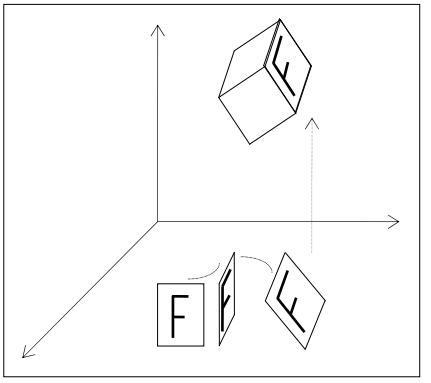


Figura 30 – Aplicarea texturii

Observaţie. La caracterele nesimetrice trebuie avută în vedere şi normala la plan, deoarece caracterele pot fi aplicate invers (ca în Figura 31).

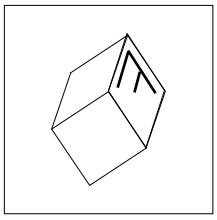


Figura 31 – Textutura aplicată invers

b) Texturile neregulate

Aceste texturi sunt utilizate în generarea aleatoare a unor forme de simulare a realităţii, de exemplu *arbori, nori, dune de nisip, valuri, lemn,* etc. Realizarea unor astfel de texturi se poate face prin:

- *Variaţia aleatoare a direcţiei de iluminare*;
- *Distorsionarea aleatoare a normalei la plan*;
- *Structuri fractale*.

Fractalii, aceste *variaţiuni geometrice ciudate*, pot fi de *formă regulată* (prin repetarea unui motiv sau detaliu primar) sau de *formă neregulată* (definiţi probabilistic).

Fractalii de *formă regulată* se definesc prin curbe, suprafeţe, volume, funcţii, etc., sau printr-o regulă de construcţie.

Fiind dată o mulţime de primitive grafice Π (corespunzătoare unei mulţimi de comenzi de desenare) cu ajutorul cărora se construieşte detaliu primar $\delta \in \Pi^*$ (o bază de plecare), se va aplica succesiv o transformare φ de n ori. Această transformare poate fi precizată printr-o funcţie, prin reguli de producţie (dacă generarea se realizează prin gramatici, aşa cum vom vedea în cele ce urmează), grafic sau printr-un subalgoritm. Desenul corespunzător descrierii obţinute reprezintă fractalul dorit .

Considerăm o mulţime de comenzi $\Pi=\{\tau_1,\tau_2,...,\tau_p\}$, conţinând primitive grafice pentru descrierea unei familii de fractali. Aceste primitive grafice pot fi desenate prin comenzile grafice corespunzătoare, deci putem obţine un desen (reprezentând un fractal) executând un astfel de şir de comenzi.

O familie de fractali cu structură regulată este mulţimea fractalilor obţinută plecând de la detaliul primar $\delta \in \Pi^*$, prin dezvoltarea pas cu pas prin regula de transformare (creştere) φ. Această regulă este aplicată fiecărei primitive grafice (comenzi de deplasare a cursorului din Π), adică $\varphi : \Pi \rightarrow \Pi^*$. Această transformare dă regula de creştere a fiecărei primitive şi este specifică fiecărei familii de fractali.

Funcţia de dezvoltare $t : \Pi^* \rightarrow \Pi^*$ care permite unui fractal să se dezvolte (să crească) în întregime cu o iteraţie (cu un *an*) este definită astfel:

$$t(\tau_1\tau_2...\tau_m) = \begin{cases} \varphi(\tau_1) & \text{dacă } m=1, \\ \varphi(\tau_1)\varphi(\tau_2)...\varphi(\tau_m) & \text{dacă } m>1. \end{cases}$$

Descrierea unui fractal după *n ani* se poate obţine aplicând funcţia de transformare de *n* ori , astfel:

$$t^n : \Pi^* \rightarrow \Pi^*, \quad t^n(w) = \begin{cases} t(w) & \text{dacă } n=1, \\ t(t^{n-1}(w)) & \text{dacă } n>1. \end{cases}$$

Putem spune că $\tau^n(\delta)$ returnează şirul de comenzi de descriere a fractalului în al *n*-lea an de dezvoltare, ceea ce înseamnă că putem defini o familie de fractali precizând mulţimea primitivelor grafice Π, detaliu primar $\delta \in \Pi^*$ şi regula de dezvoltare φ.

O *familie de fractali* de formă regulată este limbajul pictural următor :

$$\boldsymbol{F} = \{F_n(\Pi,\delta,\varphi) / n=1,2,...\},$$

unde $F_n(\Pi,\delta,\varphi) = dpic(t^n(\delta))$, adică desenul descris de Π-cuvântul $t^n(\delta)$.

Vom considera în continuare că elementele mulţimii Π sunt etichetate cu valori de la 1 la $p=|\Pi|$ ($\Pi=\{\tau_1,\tau_2,...,\tau_p\}$). În exemplele care urmează (în construcţia regulilor de transformare) vom referi funcţia *Succ*, definită în continuare:

$$\text{Succ} : \Pi \to \Pi, \quad \text{Succ}(\tau_i) = \begin{cases} \tau_{i+1} & \text{dacă } i<p, \\ \tau_1 & \text{dacă } i=p. \end{cases}$$

Considerăm mulţimea primitivelor $\Pi=\{r,u,l,d\}$, unde $(r,u,l,d) = (\rightarrow,\uparrow,\leftarrow,\downarrow)$, baza $\delta = r$ $(\rightarrow)$ şi funcţia de dezvoltare $\varphi : \Pi \to \Pi^*$, $\varphi(\tau) = \tau\ \text{Succ}(\tau)\ \tau\ \text{Succ}^{-1}(\tau)\ \tau$, $\forall \tau \in \Pi$. Familia de fractali F_1 descrisă de $t^1(\delta)$, $t^2(\delta)$, $t^3(\delta)$, ... este ilustrată în Figura 32 a.

Atât în construirea cât şi în prelucrarea fractalilor definiţi după regulile descrise anterior, este adesea utilă următoarea proprietate:

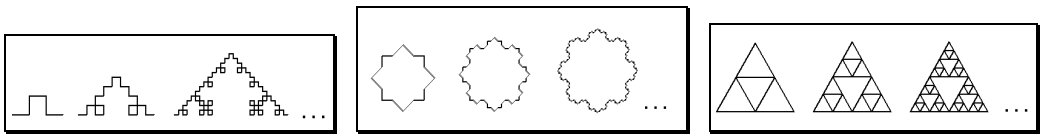
$$t^n(w_1w_2) = t^n(w_1)\ t^n(w_2), \quad \forall\ w_1,w_2 \in \Pi^* \text{ şi } n > 0.$$

Această proprietate ne permite să dezvoltăm sau să prelucrăm un fractal pe porţiuni considerându-l format din subfractali, care respectă aceeaşi regulă de dezvoltare, aplicată mai multor baze, aşa cum se poate vedea în exemplul următor.

Exemplu.

Fie $\Pi=\{r,e,u,f,l,g,d,h\}$ unde $(r,e,u,f,l,g,d,h) = (\rightarrow,\nearrow,\uparrow,\nwarrow,\leftarrow,\swarrow,\downarrow,\searrow)$, $\delta=urdl$, iar regula de creştere $\varphi(\tau) = \tau\ \text{Succ}(\tau)\ \text{Succ}^{-1}(\tau)\ \tau$.

Familia $F_2=\{\text{dpic}(t^1(\delta)), \text{dpic}(t^2(\delta)), \text{dpic}(t^3(\delta)), \ldots\}$ reprezentată în Figura 32 b, a fost realizată utilizând proprietatea anterioară: $t^n(\delta)= t^n(u)\ t^n(r)\ t^n(d)\ t^n(l)$, pentru n=1,2,3,... . Fie $\Pi=\{a,b,c\}$ unde $(a,b,c) = (\nearrow,\searrow,\leftarrow)$, $\delta=abc$ şi $\varphi(\tau) = \tau\ \text{Succ}(\tau)\ \text{Succ}^{-1}(\tau)\ \tau\ \tau$. Familia $F_3=\{\text{dpic}(t^1(\delta)), \text{dpic}(t^2(\delta)), \text{dpic}(t^3(\delta)), \ldots\}$ este ilustrată în Figura 32 c.



a) b) c)

Figura 32 - Fractali cu structură regulată

3.6.3 Lumină şi umbră

Obiectivul acestei prelucrări este de a reda o imagine cât mai realistă şi care să ofere cât mai multă informaţie.

Elementele luate în calcul sunt următoarele:

a) Sistemul de iluminare - tip, poziţie, etc.;
b) Caracteristicile suprafeţelor - reflexie transparenţă, etc.;
c) Poziţia relativă a sistemului de iluminare şi corp.

Procesul de eliminare a suprafeţelor acoperite şi de simulare a iluminării se desfăşoară simultan, adăugând fiecărei feţe un atribut de intensitate a culorii.

a) Sistemul de iluminare

Sursele luminoase pot fi :

- *punctiforme* : becuri, soare, flacără mică, etc.;
- *distribuite* : tuburi fluorescente, ferestre, etc.;
- *ambiante* : lumină care scaldă obiectele cu aceeaşi intensitate.

b) Caracteristicile suprafeţelor

Proprietăţile *de suprafaţă* ale corpurilor sunt :

- *reflexie*: zonele unui corp pot fi *mate* (care dispersează lumina, adică o reflectă după mai multe direcţii) sau *sclipitoare* (care reflectă lumina după o direcţie);
- *transparenţă* : o zonă poate fi *translucidă* (prin care trece lumina) sau *opacă* (care nu permite trecerea luminii);
- *textura* : materialul din care este construit obiectul.

c) Poziţia relativă a sistemului de iluminare şi corp

Intensitatea luminii reflectate (I_R) este dată de *Legea reflexiei cosinus* sau *Legea lui Lambert* :

$$I_R = I_I * K_{material} * cos\ \alpha$$

unde :

- I_I este intensitatea luminii incidente;
- $K_{material}$ este o constantă de material;
- α este unghiul format de normala la plan cu direcţie de iluminare (vezi Figura 33).

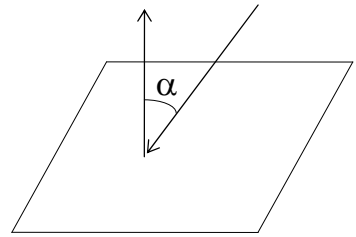


Figura 33 – Lumina reflectată

Dacă luăm în considerare şi o sursă ambiantă (I_a) atunci formula devine:

$$I_R = I_a * K^a_{material} + I_I * K_{material} * cos\ \alpha\ / (1+r^2),$$

(*r* fiind distanţa la care se află sursa, cantitatea de lumină fiind invers proporţională cu pătratul distanţei).

Iluminarea nu se face întotdeauna frontal (nu se vede întotdeauna ceea ce este şi iluminat). Se obţin efecte dacă iluminarea este nefrontală, pentru că în această situaţie (când se precizează pe lângă poziţia observatorului şi poziţia sursei de iluminare) există posibilitatea ca părţi vizibile ale corpului să nu fie iluminate (sunt umbrite). În această situaţie, calculele de *vizibilitate* (prezentate la *eliminarea suprafeţelor nevăzute*) se vor face atât din poziţia observatorului (Ω) cât şi din poziţia sursei de iluminare (S). Pentru fiecare pixel vom avea următoarele situaţii:

- este *văzut* atât de Ω cât şi de S - este *vizibil* şi *luminat*;
- nu este *văzut* atât de Ω - este *invizibil* , deci nu se desenează;
- este *văzut* atât de Ω dar nu şi de S - este *vizibil* dar *umbrit*.

Pentru reprezentare se poate folosi un algoritm de tip *z-buffer* modificat sau *metoda drumului optic* (*Ray-Tracing* descrisă în [[63]).